November 1993
Vol. 3 No. 11

INSIDE HYPERCARD

Tips & techniques for HyperCard and HyperTalk

Building an amortization schedule

by Ron Wilder

If you've ever taken out a loan to buy a new car or house, you probably realize that even with a low interest rate, you end up paying a large amount of money in interest over the life of the loan. If you really want to torture yourself, you can look at how much of your monthly payment goes to interest and compare it to the amount the bank applies to the principal. To see the precise breakdown between principal and interest at each point in the life of your loan, you'll need to obtain an amortization schedule.

Historically, lending institutions painstakingly prepared amortization schedules by hand. Today, they use computer programs to prepare the schedules. Although most institutions use spreadsheets to prepare amortization schedules, you can create an amortization schedule in HyperCard in about the same length of time it would take to enter the needed formulas into a spreadsheet. In this article, we'll show you how to create your own amortization schedule stack.

How the stack works

An amortization schedule displays the amounts of interest and the applied principal for each payment. Therefore, before we can build an amortization schedule, we must know the loan amount, the term of the loan, and the interest rate of the loan.

We'll then create a script that calculates the monthly payment along with the breakdown of interest and principal applied to each payment. HyperCard will determine these amounts for the life of the loan.

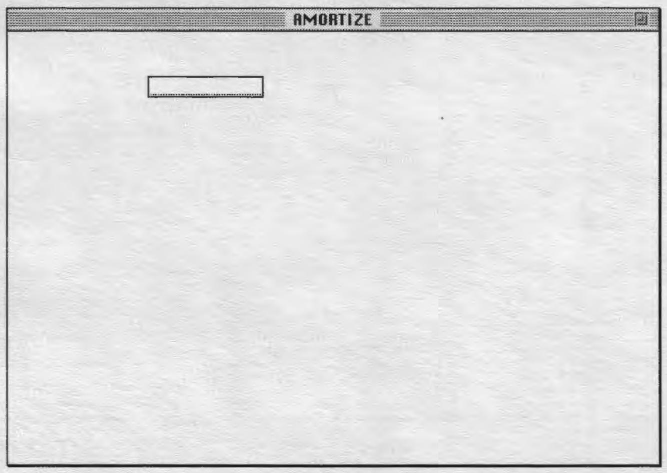
Let's begin by creating the stack and the card fields. Then, we'll insert handlers into the appropriate locations.

Building an amortization table stack

Begin by opening a new stack or navigating to an empty card. Then, choose New Field from the Objects menu. Double-click on the field to open its Field Info dialog box. Type *LoanAmount* into the Field Name text box, and then press [return] to dismiss the dialog box. Resize the field and position it in the upper-left corner of the screen, as shown in Figure A.

Again, choose New Field from the Objects menu. Double-click on the field to open its Field Info dialog box. Type *Year* into the Field Name text box and press [return] to dismiss the dialog box. Resize the Year field to match the LoanAmount field, and then position the second field directly below the first.

Figure A



Resize the field to make it a single-line field and then move it into the upper-left corner of the card.

Now, create another field and double-click on it to open the Field Info dialog box. Type *Rate* into the Field Name text box and press [return] to dismiss the dialog box. Then, resize the field to match the LoanAmount and Year fields and position it below those two fields.

Finally, create another field and double-click on it to open its Field Info dialog box. Type *Payment* into the Field Name text box and press the [return] key to dismiss the dialog box. Now, resize and position it as you did the

IN THIS ISSUE

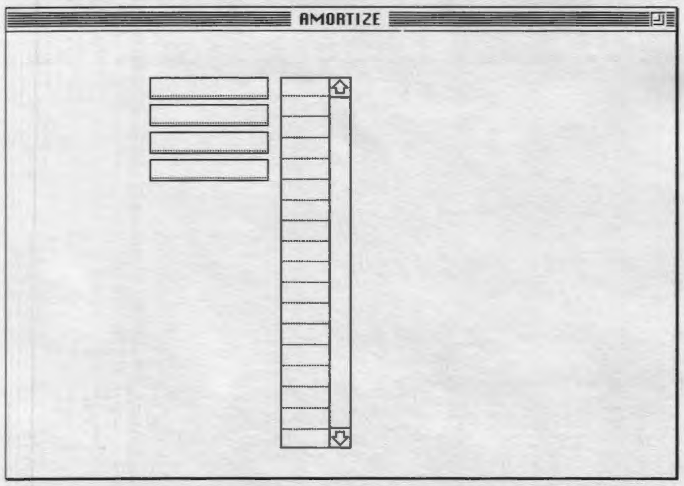
A Publication of The Cobb Group

other fields. You're now ready to create the four fields that will store the monthly payment information.

Defining the report area

To create the amortization schedule itself, choose New Field from the Objects menu, and then double-click on the field to open its Field Info dialog box. Type *Number* into the Field Name text box and select Scrolling from the Style options. Then, select the Lock Text option. Finally, press [return] to dismiss the dialog box. Now, resize the field to accommodate at least twelve lines and position it in the middle of the card, as shown in Figure B.

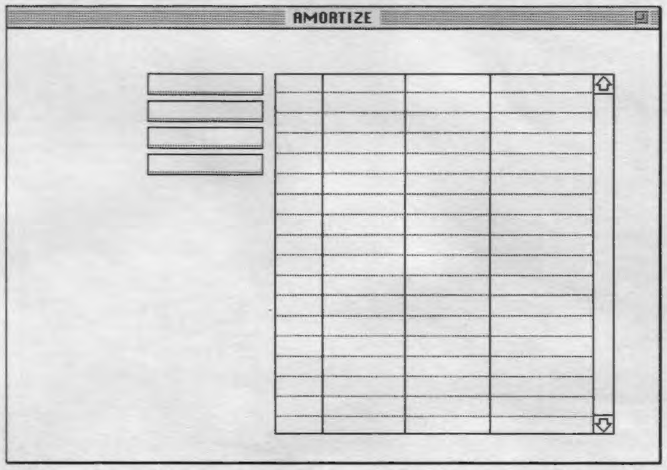
Figure B



Resize the Number field to allow for several entries and then position it in the middle of the card.

Use the same procedure to create the three scrolling fields *Interest*, *Principal*, and *PrincipalBalance*. Be sure to activate the Lock Text option for each field. When you place these three fields on the card, position them so the left edge of each field covers the scroll bar of the preceding field, as shown in Figure C.

Figure C



Position the fields so that only the last scroll bar is visible.

Next, you'll want to identify the fields with labels. To do so, select the Text tool (A) from the Objects menu and click to the left of the LoanAmount field. Type *Loan Amount*. Then, click to the left of the Year field and type *Years of Loan*. Next, click to the left of the Rate field and

INSIDE HYPERCARD

Inside HyperCard (ISSN 1052-9470) is published monthly by The Cobb Group.

Prices	Domestic	$59/year ($7.00 each)
	Outside U.S.	$79/year ($8.50 each)
Phone	Toll free	(800) 223-8720
	Local	(502) 491-1900
	Customer Relations Fax	(502) 491-8050
	Editorial Department Fax	(502) 491-4200

Address
You may address tips, special requests, and other correspondence to

The Editor, *Inside HyperCard*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster
Second class postage is paid at Louisville, KY.
Send address changes to

Inside HyperCard
P.O. Box 35160
Louisville, KY 40232

Authorized Canada Post International Publications Mail (Canadian Distribution) Sales Agreement #XXXXXX CANADA GST #123669673. Send returns to Canadian Direct Mailing Sys. Ltd., 920 Mercer Street, Windsor, Ontario N9A 7C2. Printed in the USA.

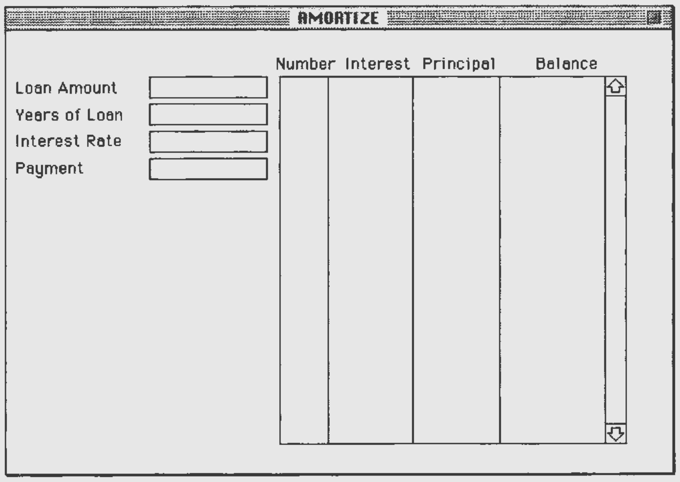
Staff

Editor-in-Chief	Michael Stephens
Assistant Editor	Ron Wilder
Editing	Meredith Little
	Linda Recktenwald
	Tiffany Taylor
Production Artists	Tre Pryor
	Laurie A. Wolberton
Design	Karl Feige
Publications Manager	Maureen Spencer
Managing Editor	Michael Stephens
Circulation Manager	Marjorie Glassman
Publications Director	Linda Baughman
Editorial Director	Jeff Yocom
Publishers	Mark Crane
	Jon Pyles

type *Interest Rate*. Finally, click to the left of the Payment field and type *Payment*.

Now, click directly above the Number field and type *Number*. Press the [spacebar] until the cursor appears over the Interest field and type *Interest*. Continue this process to label the Principal and Balance fields. Your stack should now look similar to the one shown in Figure D. You're now ready to enter the stack's handlers.

Figure D



Your stack should now contain eight fields.

Creating the calculation handler

The amortization stack's handlers tell HyperCard to create an amortization schedule based on the data you enter into the loan information fields. To create the schedule, you'll create a calculation button that performs the necessary calculations.

To create the calculation button, you choose New Button from the Objects menu. Then, double-click the new button to open its Button Info dialog box. Type *Calculate* in the Button Info text box. Then, click Script… to open the button's script, and enter this handler:

```
on mouseUp
  set numberFormat to "0.00"
  put card field "LoanAmount" into¬
  LoanAmount
  put card field "Year" into Year
  put card field "Rate"/100 into Rate
  put Rate/12 into InterestRate
  put Year*12 into Periods
  put (InterestRate*LoanAmount)/¬
  (1-(InterestRate+1)^-Periods)¬
  into Payment
  put Payment into card field "Payment"
  set cursor to watch
  put LoanAmount into PrincipalBalance
  repeat with Number=1 to Periods
    put PrincipalBalance * InterestRate¬
    into Interest
    put Payment-Interest into Principal
    if Number=Periods then put¬
    PrincipalBalance into Principal
    put PrincipalBalance-Principal¬
    into PrincipalBalance
    put Interest & return after¬
    line Number of InterestTemp
    put Principal & return after¬
    line Number of PrincipalTemp
    put PrincipalBalance & return¬
    after line Number of¬
    PrincipalBalanceTemp
  end repeat
  lock screen
  put InterestTemp into card¬
  field "Interest"
  put PrincipalTemp into card¬
  field "Principal"
  put PrincipalBalanceTemp¬
  into card field "PrincipalBalance"
  set numberFormat to "0"
  repeat with V=1 to Periods
    put V & return after line V¬
    of Vtemp
    put Vtemp into card field¬
    "Number"
  end repeat
  unlock screen
  set cursor to arrow
end mouseUp
```

Save the script and close the Script Editor to return to the card. Now you're ready to complete the stack by entering the stack handlers.

Creating the stack handlers

To complete the stack, you'll want to have the stack clear the previous entries in the fields. You'll also want to have the amortization schedule fields scroll simultaneously. You accomplish both of these tasks by placing handlers in the stack script.

To construct the stack handlers, select Stack Info… from the Objects menu. In the Stack Info dialog box, click on the Script button to open the stack's script, and enter these handlers:

```
on openStack
  put empty into field "Year"
  put empty into field "LoanAmount"
  put empty into field "Rate"
  put empty into field "Payment"
  put empty into field¬
  "PrincipalBalance"
  put empty into field "Principal"
  put empty into field "Interest"
  put empty into field "Number"
end openStack

on mouseWithin
  set the scroll of card field¬
  "Number" to scroll of card¬
  field "PrincipalBalance"
  set the scroll of card field¬
  "Principal" to scroll of card¬
  field "PrincipalBalance"
  set the scroll of card field¬
  "Interest" to scroll of card¬
  field "PrincipalBalance"
end mouseWithin
```

Save the script and close the Script Editor to return to the stack. Now, choose the Browse tool from the Tools palette. You're now ready to use the stack.

Using the amortization stack

Suppose you just bought a brand new Mazda Miata for $19,459 at 9% interest for 5 years. To build the amortization schedule, you first enter *19459* into the field labeled Loan Amount. Then, open the field labeled Years of Loan and enter *5*. Finally, enter *9* into the Interest Rate field.

To build the schedule, you simply click the Calculate button. When you do so, HyperCard calculates the Payment to be $403.94, the mouse pointer changes to the watch cursor, and HyperCard calculates the amortization schedule. When the script completes the schedule, the watch cursor changes back into the mouse pointer.

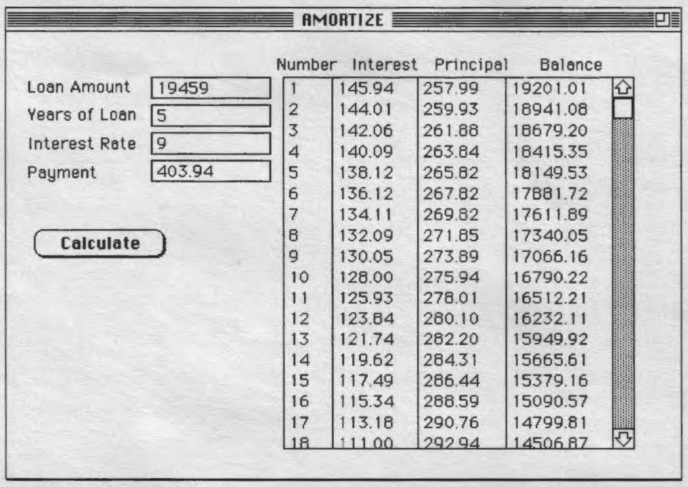
From the amortization schedule shown in Figure E, you can see that $145.94 of the first $403.94 payment is applied to interest, and the balance after the first payment is $19,201.01. You'll also notice that as payments are made, the interest payments decrease and the principal payments increase.

Going one step further

With the basic financial mathematics included in the script, you can now improve on both the features and the function of the amortization stack. For instance, you could create an additional field that allows you to build amortization schedules on quarterly payments.

You could also create a field that represents occasional additional payments that apply directly to the principal. In doing so, you'll discover that this is an excellent means of decreasing your overall interest payment.

Figure E



Loan Amount	19459
Years of Loan	5
Interest Rate	9
Payment	403.94

Number	Interest	Principal	Balance
1	145.94	257.99	19201.01
2	144.01	259.93	18941.08
3	142.06	261.88	18679.20
4	140.09	263.84	18415.35
5	138.12	265.82	18149.53
6	136.12	267.82	17881.72
7	134.11	269.82	17611.89
8	132.09	271.85	17340.05
9	130.05	273.89	17066.16
10	128.00	275.94	16790.22
11	125.93	278.01	16512.21
12	123.84	280.10	16232.11
13	121.74	282.20	15949.92
14	119.62	284.31	15665.61
15	117.49	286.44	15379.16
16	115.34	288.59	15090.57
17	113.18	290.76	14799.81
18	111.00	292.94	14506.87

The calculated payment is $403.94 per month.

Conclusion

Reviewing an amortization schedule is a great way to keep an eye on the amount of principal that your bank is applying to the loan with each payment. In this article, we've shown you how to create a stack that builds a basic amortization schedule. ▩

ANIMATION TECHNIQUE

Adding incremental animation to your stacks

In previous issues of *Inside HyperCard*, we've discussed a number of animation techniques. Because each stack poses its own set of difficulties, the more tricks you have in your repertoire, the better. The animation technique we'll present in this article works especially well when you need to move card or background objects across the screen precisely and smoothly. Since the technique doesn't require you to switch tools, as do click-and-drag animation sequences, it may operate more quickly in some circumstances. Now, let's use incremental animation to add a little life to a stack.

What is incremental animation?

When you animate an object, you essentially tell HyperCard to move that object along a series of screen coordinates. Incremental animation uses a repeat loop to advance in regular increments the coordinates the object will follow. This animation technique works best with objects that have clearly defined borders, such as fields or buttons.

To animate an object, you simply identify the object's location with one of the location properties. If you need only to create vertical or horizontal movement, you can use the `top`, `bottom`, `right`, or `left` property. If you need to create diagonal movement, you'll need to use the `loc`, `topLeft`, or `botRight` property.

After you identify the object's location, you decide how you want to move it. For example, if you add to an object's `top` or `bottom` value, the object moves down on the screen; subtracting from either value moves the object up on the screen. Similarly, adding to the `right` or `left` value moves the object to the right. Subtracting from `right` or `left` moves the object to the left.

If you want to create diagonal movement, you'll have to simultaneously alter the h and v components of the `loc`, `topLeft`, or `botRight` property. Now, let's create a simple script to demonstrate how to use incremental animation.

Using incremental animation

Begin by opening a new stack. Then, choose New Button from the Objects menu. Double-click the new button to open its Button Info dialog box. Type *Moving* in the Button Name text box. Click Script... to open the button's script.

Enter the following handler for the Moving button:

```
on mouseUp
  repeat with x = 300 down to 25
    set the top of me to x
  end repeat
  wait until the mouseClick
  repeat with x = 25 to 300
    set the top of me to x
  end repeat
end mouseUp
```

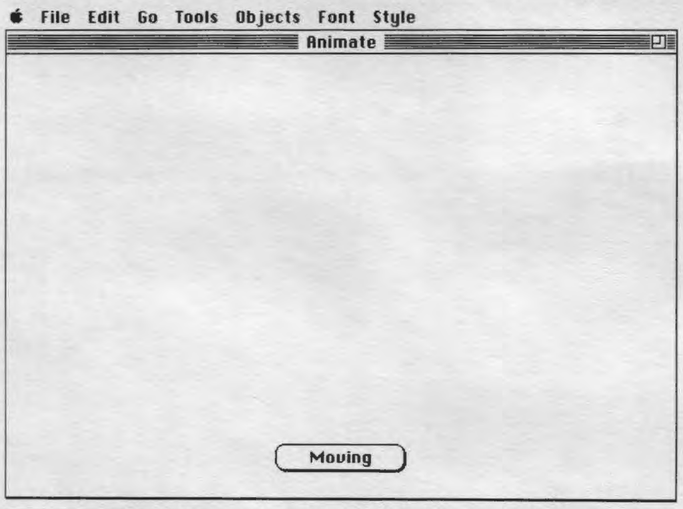
Save the script and close the Script Editor to return to the card.

Next, choose the Browse tool from the Tools palette. To reposition the button, open the Message box and type

```
set the top of card button 1 to 300
```

When you press [return], HyperCard places the Moving button at the bottom center of your card, as shown in Figure A.

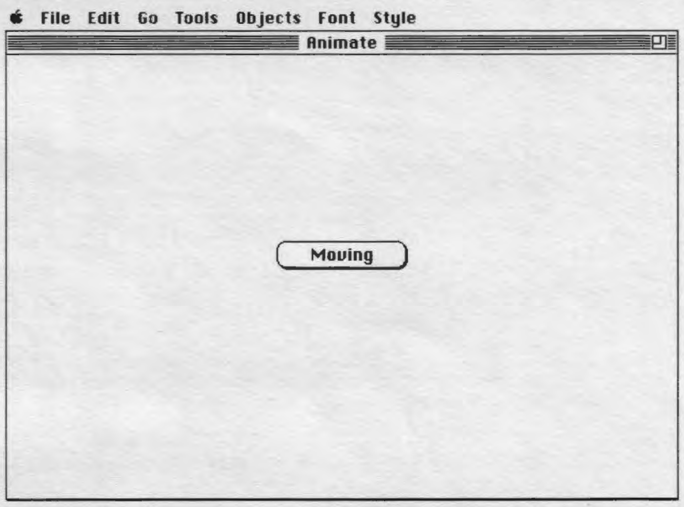
Figure A



HyperCard moves the Moving button to the bottom center of your card.

To demonstrate the Moving button's incremental animation sequence, simply click the button. As Figure B shows, the Moving button moves upward on the screen. When it reaches the top of the screen, the button waits until you click the mouse button before returning to the bottom.

Figure B



The Moving button moves upward on the screen.

Creating diagonal motion

As we mentioned earlier, you can also use this technique to produce diagonal motion. To do so, you must simultaneously change the vertical and horizontal components of the object's direction. Now, let's modify our vertical motion script to produce diagonal motion.

Begin by holding down the ⌘ and [option] keys and clicking the Moving button to open its script. Then, replace the current handler with this one:

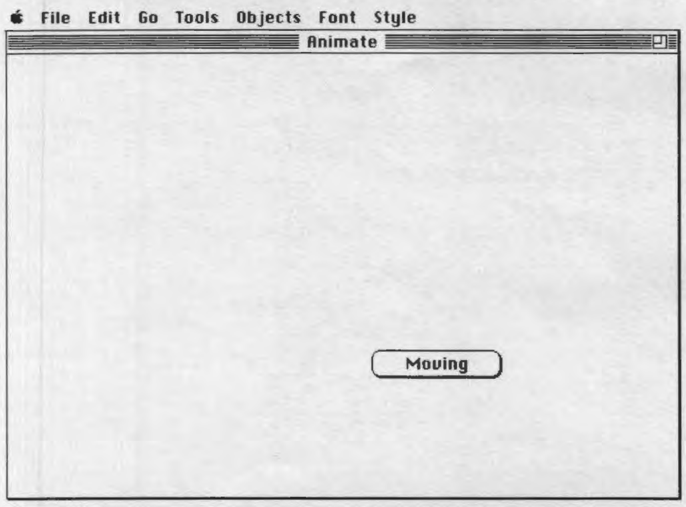
```
on mouseUp
  put item 1 of the loc of me into h
  put item 2 of the loc of me into v
  repeat until h ≥ 450 or v ≤ 10
    set the loc of me to h,v
    add 1 to h
    subtract 1 from v
  end repeat
end mouseUp
```

Save the script and close the Script Editor to return to the card.

To test the Moving button's new script, simply click the button. As Figure C shows, the button moves diagonally across the screen until it reaches the right border of the card.

You can change the angle and direction at which the button travels, by altering the increments that the script uses to advance the h and v components of the button's position. For example, let's change the script again so that the button travels diagonally in another direction. We'll also make the button travel at a different angle.

Figure C



The Moving button travels diagonally across the screen.

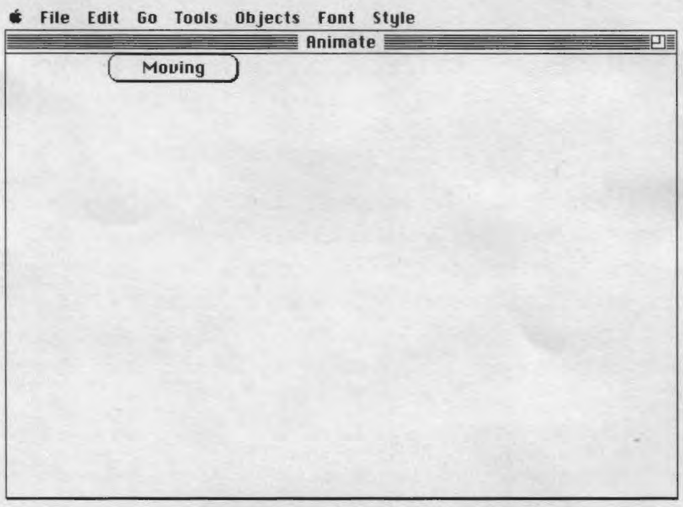
Hold down the ⌘ and [option] keys and click the Moving button to open its script. Then, replace the existing handler with this one:

```
on mouseUp
  put item 1 of the loc of me into h
  put item 2 of the loc of me into v
  repeat until h ≥ 450 or v ≤ 10
    set the loc of me to h,v
    subtract 3 from h
    subtract 1 from v
  end repeat
end mouseUp
```

Save the script and close the Script Editor to return to the card.

As Figure D shows, when you click the button, it moves up and to the left. This time, the button travels horizontally three pixels for each pixel it climbs vertically.

Figure D



The button travels diagonally in a different direction.

Notes

You also can use this technique to move a field. However, you must first lock the text of the field if you want the motion handlers to reside in the field's script. An unlocked field will not recognize the `mouseUp` message. If you want to move an unlocked field, you can activate the handler from an external source, such as a button, card, or stack handler.

Conclusion

Animation can add a little life to your stacks. HyperCard provides a number of ways to animate text, graphics and stack objects. In this article, we showed you a technique called incremental animation. You can use incremental animation to produce fluid motion along a straight line. ■

BUILDING A BETTER USER INTERFACE

Creating pop-up fields

One of the best ways to conserve space in your stacks is to create objects that expand when you're using them and then contract to a smaller size when you no longer need them. Text fields are one of the best candidates for this type of approach. If you create text fields that expand only when you need them, you can include a lot of information in a stack without it occupying a lot of space.

In this article, we'll show you how to create pop-up fields. Along the way, we'll discuss a couple of alternatives for employing these fields in your stacks.

How pop-up fields work

The pop-up fields we'll create expand when you double-click on them. The fields contract to their original size when you single-click on them. To expand the field, you simply set the field's `rect` property to coordinates large enough to encompass the data you want to present. The field contracts to a single line of text.

The field's script uses a simple conditional expression to determine whether the user double-clicks or single-clicks. When the field receives the first click, it reads the value of the `ticks` function. When the user clicks again, the handler again reads the **ticks** function. If 2 clicks occur within 20 ticks, the handler considers the action to be a double click and expands the field.

If the second click occurs after 20 ticks, the handler considers it to be a separate action and contracts the field. Now, let's use this technique to create a pop-up field.

Building the field

Begin by opening a new stack or by navigating to an empty card. Next, choose New Field from the Objects menu. Double-click the new field to open its Field Info dialog box.

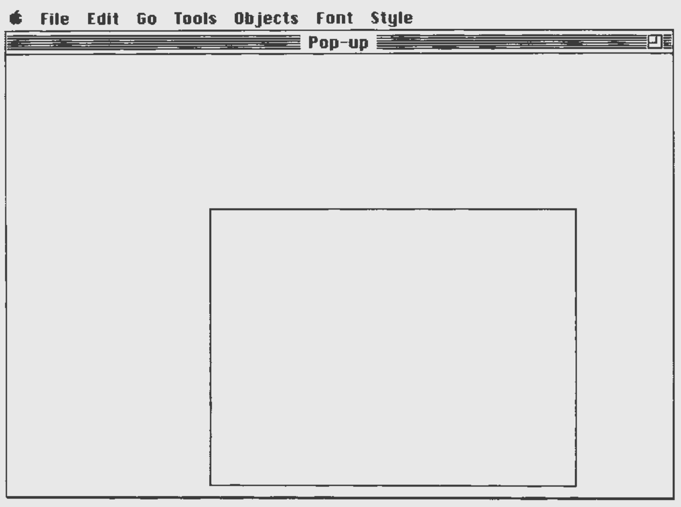
Type *Pop-up* in the Field Name text box. Then, click Script… to open the field's script. Enter the following handler for the Pop-up field:

```
on mouseUp
  put the ticks into old
  wait until the mouseClick or¬
  the ticks-old>20
  if the ticks-old<20 then
    set the rect of me to 10,10,250,250
  else
    set the rect of me to 10,10,200,30
  end if
end mouseUp
```

Save the script and close the Script Editor to return to the card.

Next, place the mouse pointer on the lower-right corner of the field, hold down the mouse button, and drag down and to the right to expand the field. Then, choose the Browse tool from the Tools palette. At this point, your card should look like the one shown in Figure A.

Figure A



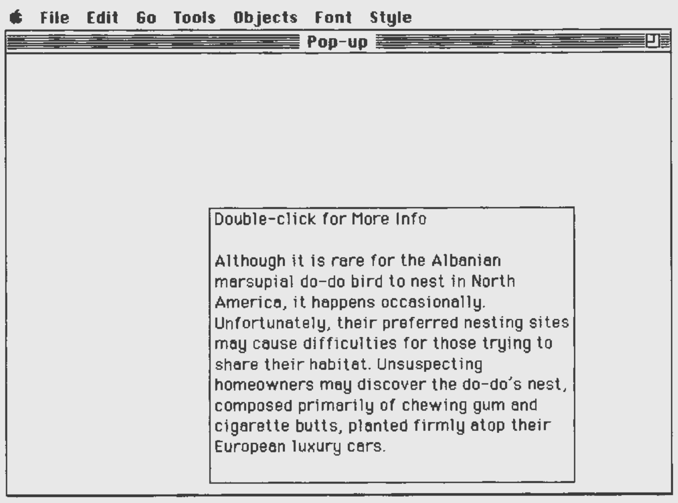
Drag the lower-right corner of the field to expand it.

Next, click in the upper-left corner of the field. Type *Double-click for More Info* and press [return] twice. Then, type the following paragraph:

Although it is rare for the Albanian marsupial do-do bird to nest in North America, it happens occasionally. Unfortunately, their preferred nesting sites may cause difficulties for those trying to share their habitat. Unsuspecting homeowners may discover the do-do's nest, composed primarily of chewing gum and cigarette butts, planted firmly atop their European luxury cars.

At this point, your card should look like the one shown in Figure B.

Figure B



Double-click for More Info

Although it is rare for the Albanian marsupial do-do bird to nest in North America, it happens occasionally. Unfortunately, their preferred nesting sites may cause difficulties for those trying to share their habitat. Unsuspecting homeowners may discover the do-do's nest, composed primarily of chewing gum and cigarette butts, planted firmly atop their European luxury cars.

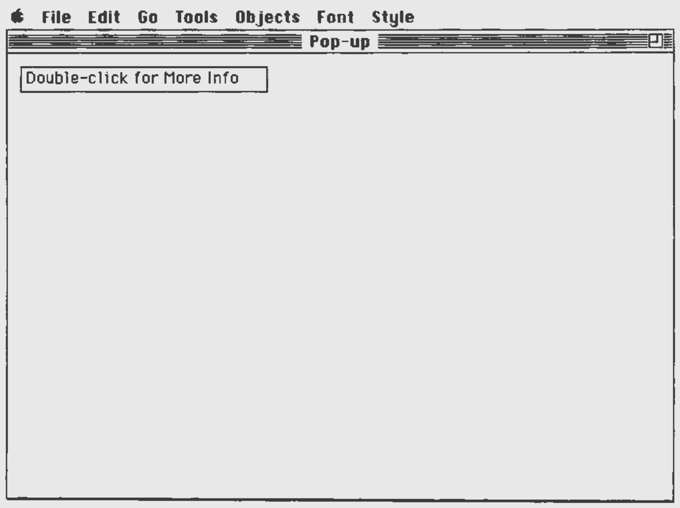
Enter this text into the Pop-up field.

Next, open the Message box and enter the command

```
set the lockText of card field "Pop-up" to true
```

to lock the field. Then, click on the field to contract and reposition it, as shown in Figure C.

Figure C

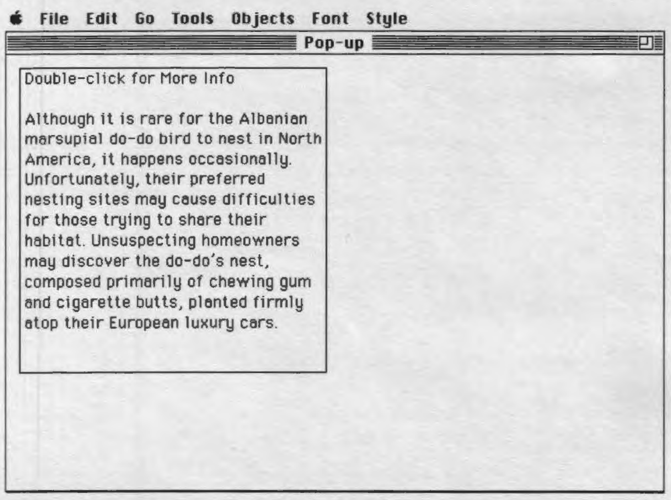


Double-click for More Info

Click on the field to contract it and reposition it.

To use the Pop-up field, simply double-click on it. As Figure D shows, when you double-click on the field, it expands so you can read its text. If you click on the field, it contracts again. Now, let's look at a way to save even more space with pop-up fields.

Figure D



Double-click for More Info

Although it is rare for the Albanian marsupial do-do bird to nest in North America, it happens occasionally. Unfortunately, their preferred nesting sites may cause difficulties for those trying to share their habitat. Unsuspecting homeowners may discover the do-do's nest, composed primarily of chewing gum and cigarette butts, planted firmly atop their European luxury cars.

The Pop-up field expands when you double-click on it.

Creating scrolling pop-up fields

In some circumstances, you may not want the field to expand enough to display all the field's text. In these cases, you can change the field into a scrolling field when it expands. To demonstrate this, let's modify our pop-up field.

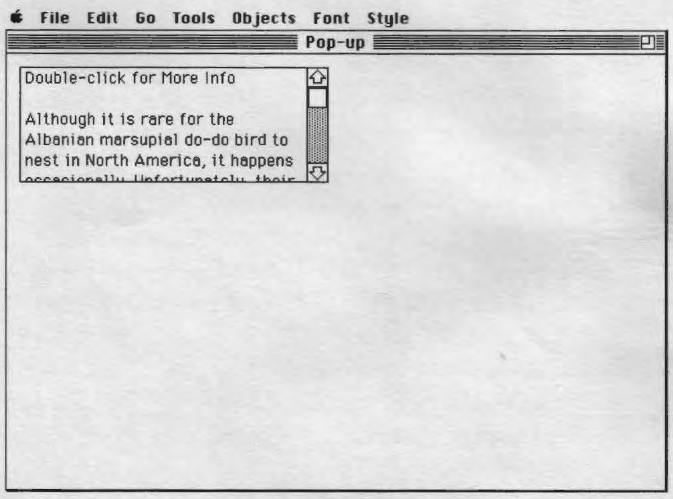
First, hold down the [shift], ⌘, and [option] keys and click on the Pop-up field to open its script. Then, replace the existing handler with this one:

```
on mouseUp
  put the ticks into old
  wait until the mouseClick or¬
  the ticks-old>20
  if the ticks-old<20 then
    set the rect of me to 10,10,250,100
    set the style of me to scrolling
  else
    set the rect of me to 10,10,200,30
    set the style of me to rectangle
  end if
end mouseUp
```

Save the script and close the Script Editor when you finish.

To demonstrate this handler, simply double-click on the Pop-up field. As Figure E shows, although the expanded field is smaller, it includes a scroll bar that you can use to access the information not displayed onscreen. To contract the field, simply click anywhere on its text.

Figure E



Double-click for More Info

Although it is rare for the Albanian marsupial do-do bird to nest in North America, it happens

After you modify its script, the expanded Pop-up field includes a scroll bar.

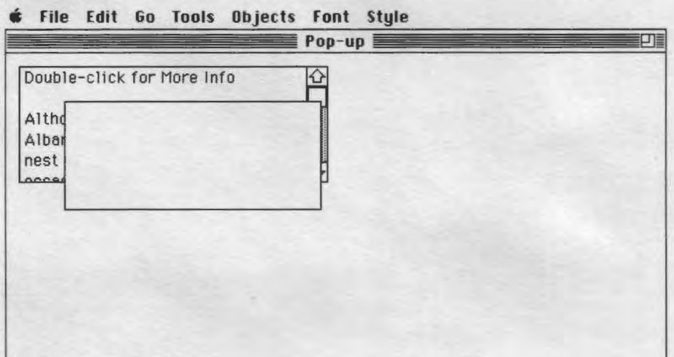
A potential problem and how to avoid it

Although creating pop-up fields is a pretty straightforward process, you may run into a problem when you place them in your stacks. HyperCard establishes an order of precedence when you create card or background objects. In effect, this means that HyperCard places the most recently created objects in layers on top of objects created earlier. For this reason, other card objects may obscure portions of your pop-up fields.

To demonstrate this, choose New Field from the Objects menu. Then, drag the new field and place it directly below the Pop-up field. Next, choose the Browse tool from the Tools palette.

Now, double-click on the Pop-up field. As Figure F shows, the new field appears on top of the Pop-up field.

Figure F



Double-click for More Info

Alth
Alba
nest

The new field obscures the Pop-up field.

Fortunately, it's easy to correct this situation. You can avoid the problem altogether by simply creating the pop-up fields after you create the other elements of your stack. By creating the pop-up fields last, you ensure that they'll take precedence over the other card or background objects.

If you need to change an object's order of precedence, you can use the Bring Closer and Send Farther commands on the Objects menu. You simply select the pop-up field and choose Bring Closer until the field resides in the top layer. For more information about changing the precedence of HyperCard objects, see "Changing the Order of Precedence for HyperCard Objects," in the January 1993 issue of *Inside HyperCard*.

Conclusion

HyperCard provides a number of ways to present text-based information. Unfortunately, large blocks of text may occupy more space on your card than you'd like. By creating pop-up fields like the ones we describe in this article, you can conserve space without compromising the quantity of information you present.

HYPERPROJECT

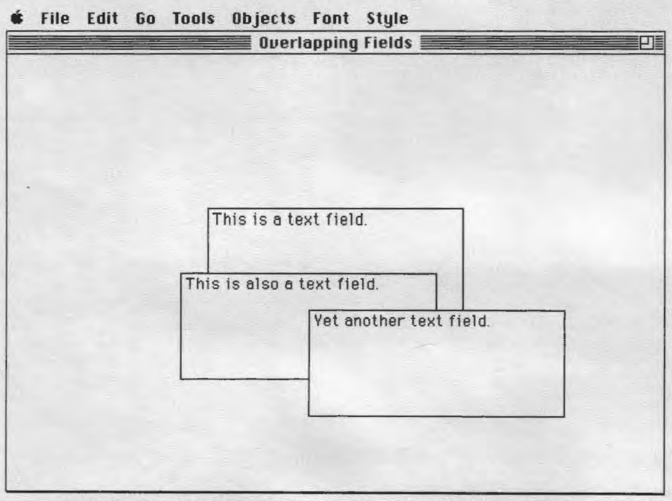
Palettizing a HyperCard field

We based this article on an idea submitted by Robert McBurney. To thank Mr. McBurney for his contribution, we're sending him a check for $25.

HyperCard provides a couple of ways to display text in your stacks. You can put the text in a field either in the background or card layer of your stack. You can also use the Text tool to paint the text directly onto the card or the background. Unfortunately, if you use either of these methods, your users can't relocate the text easily when they're using your stack in Browse mode.

Putting text in fields or directly on cards can cause another problem as well. As Figure A shows, when you overlap fields, the most recently created fields obscure those below them. Although the text on the lower fields remains, you can't see it unless you select the Field tool and reposition the fields.

Figure A



More recently created fields obscure other fields when they overlap on a card.

Fortunately, HyperCard provides a solution. If you place your text in a palette, your users will be able to reposition the palette onscreen as easily as they'd move any other desktop window. In this article, we'll show you a script that turns a card or background field into a palette.

The technique

You'll need to do several things in order to palettize a text field. First, you'll create the appropriate handlers in your Home stack. Then, you'll use the Palette Maker card in the Power Tools stack to create a palette that activates those handlers. While you're using the Power Tools stack, you'll also use the Resource Mover to add the `newPalette` resource to the Home stack. Finally, you'll need to use the `palette` external command to display your new palette.

The scripts we'll create use the `newPalette` external command. We'll also use the Export Paint and Import Paint commands. If you're unfamiliar with the Import Paint and Export Paint commands, see "Exporting and Importing a Snapshot of Your Stack," on page 14. Now, let's begin by placing the palettizer handlers in the Home stack's script.

Creating the palettizer handlers

We'll add the palettizer scripts to the end of the Home stack's stack script. We'll enter two handlers—one for the `palettizeIt` message and one for the `createPalette` message. We'll later create a palette that generates the `palettizeIt` message. The `palettizeIt` handler generates the `createPalette` message.

Begin by opening your Home stack. Next, choose Stack Info... from the Objects menu. Click Script... in the Stack Info dialog box to display the Home stack's script. Next, scroll to the end of the script and enter the following handlers:

```
-------palettizer handlers-----

on palettizeIt
  global fnum,tl,br
  push card
  if the selectedField is not empty then
    put the selectedField into fnum
    repeat with x = 1 to the number of¬
      card fields
      hide card field x
    end repeat
    repeat with x = 1 to the number of¬
      background fields
      hide background field x
    end repeat
    show fnum
    put the topLeft of fnum into tl
    put the botRight of fnum into br
    choose select tool
    export paint to file fnum
    doMenu "New Card"
    import paint from file fnum
    createPalette
    choose select tool
    domenu "Select All"
    domenu "Cut Picture"
    choose browse tool
  end if
  doMenu "Delete Card"
  repeat with x = 1 to the number of¬
    card fields
    show card field x
  end repeat
  repeat with x = 1 to the number of¬
    background fields
    show background field x
  end repeat
  pop card
end palettizeIt

on createPalette
  global paLID,fnum,tl,br
  put 2052 into paLID
  lock screen
  choose select tool
  -- get the rect of fnum
  drag from tl to br
  doMenu "Copy Picture"
  choose browse tool
 newPalette fnum,paLID,0,0,0,"",""
  palette fnum
end createPalette

-- To use these handlers, you must¬
-- install the Palletizer! palette
-- in the Home stack.
```

Save the script and close the Script Editor to return to the Home stack.

The `palettizeIt` handler begins by identifying the selected field. Next, it hides all the unselected fields. Then, the script uses the `export paint to file` command to create a bitmap image of the card. The script imports that bitmap image and issues the `createPalette` message. At this point, HyperCard executes the `createPalette` handler.

The `createPalette` handler uses the Select tool to copy the portion of the card that contains the text field. Then, the handler executes the `newPalette` external command and returns control to the `palettizeIt` handler.

The `palettizeIt` handler restores the card to its original state and returns control to the user. When the handlers terminate, the text of the selected field appears on a palette floating above the card.

After you create these handlers, you must build the Palettizer! palette. To do so, you'll need to use the Palette Maker card in the Power Tools stack.

Creating the Palettizer! palette

The Palette Maker card can be a little confusing if you haven't worked with it before. The Palette Maker card allows you to place buttons and artwork together in a resizable work area. Then, the card's scripts automatically convert the buttons and artwork into a palette.

To use the Palette Maker, we'll first create a new palette card. Then, we'll name the palette and set up its characteristics. Finally, we'll create the buttons and artwork for the Palettizer! palette.

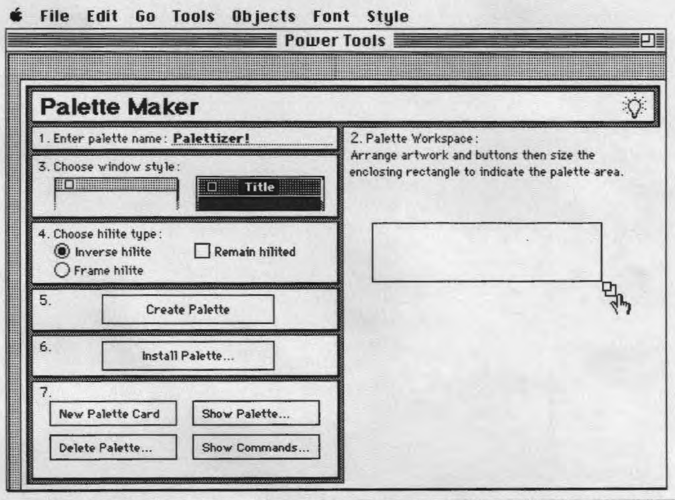
To access the Power Tools stack, type

```
go "Power Tools"
```

into the Message box and press [return]. HyperCard should take you to the first card of the Power Tools stack. Click on the words Palette Maker to move directly to the Palette Maker card. Next, click the New Palette Card button to create a new palette workspace.

Now, place the tip of the mouse pointer in the palette work area's resize box. Hold down the mouse button and drag the resize box until the palette work area looks like the one shown in Figure B.

Figure B



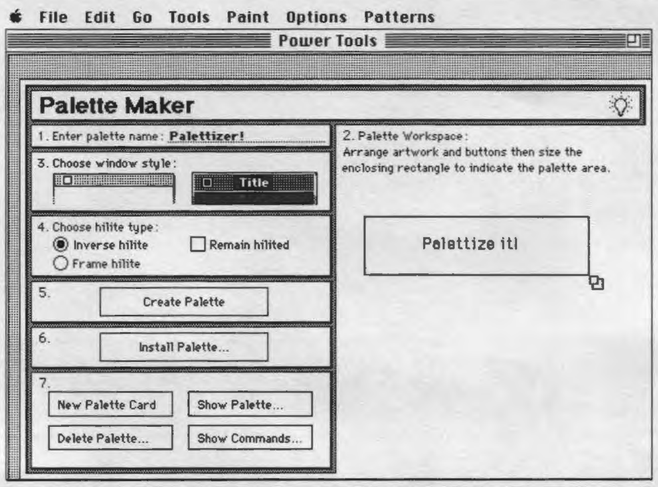
1. Enter palette name: Palettizer!
2. Palette Workspace:
Arrange artwork and buttons then size the enclosing rectangle to indicate the palette area.
3. Choose window style:
4. Choose hilite type:
[✓ Inverse hilite] ☐ Remain hilited
Frame hilite
5.
6.
7.

Use the resize box to resize the palette work area.

Next, type *Palettizer!* into the Enter Palette Name text box. Then, select the Title option from the Choose Window Style options. Select Inverse Hilite from the Choose Hilite Type options. Deactivate the Remain Hilited option if it is active.

Now, choose the Text tool from the Tools palette. Click in the palette work area and type *Palettize it!* as shown in Figure C. When you finish, choose the Browse tool.

Figure C



1. Enter palette name: Palettizer!
2. Palette Workspace: Arrange artwork and buttons then size the enclosing rectangle to indicate the palette area.
3. Choose window style:
4. Choose hilite type: (●) Inverse hilite ( ) Frame hilite [ ] Remain hilited

Palettize it!

Select the Text tool and type Palettize It! *into the palette work area.*

Now, choose New Button from the Objects menu. Drag the new button over the words Palettize it! in the palette work area. Double-click the new button to open its Button Info dialog box. Type *Palettize it!* into the Button Name text box. Then, click Script... to open the button's script.

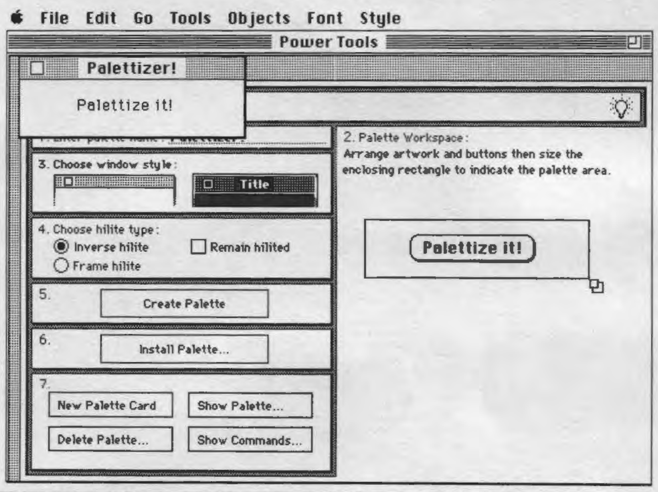
Type the following handler for the Palettize it! button:

```
on mouseUp
  palettizeIt
end mouseUp
```

Save the script and close the Script Editor to return to the Palette Maker card. Then, choose the Browse tool from the Tools palette.

To create the palette, click the Create Palette button. As Figure D shows, HyperCard will create and display the Palettizer! palette.

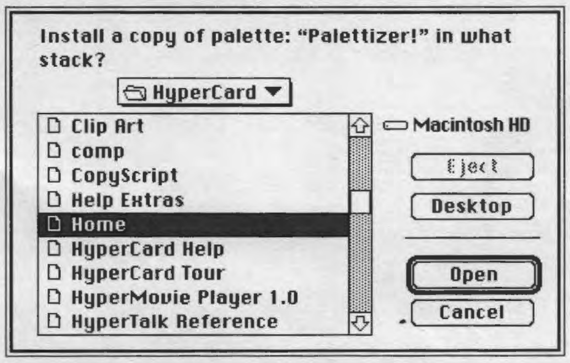
Figure D



HyperCard generates the Palettizer! palette.

Now, you need to install this palette in the Home stack. To do so, click the Install Palette... button. When HyperCard presents the Install Palette dialog box, navigate to the Home stack, as shown in Figure E. Double-click on the Home stack's name in the navigation box to install the palette. HyperCard will present the Installation Done dialog box when it finishes.

Figure E



Install a copy of palette: "Palettizer!" in what stack?

Select the Home stack in the Install Palette dialog box.

Moving the newPalette resource

Before you leave the Power Tools stack, you need to use the Resource Mover to install the `newPalette` XCMD into the Home stack. Fortunately, the Resource Mover is very easy to use. After you navigate to the Resource Mover card you'll open the Power Tools stack in the list box on the left side of the card and open the Home stack on the right side of the card. Then, you'll select the `newPalette` resource from the list box showing the resources in the Power Tools stack. Finally, you'll click Copy to install the resource in the Home stack.

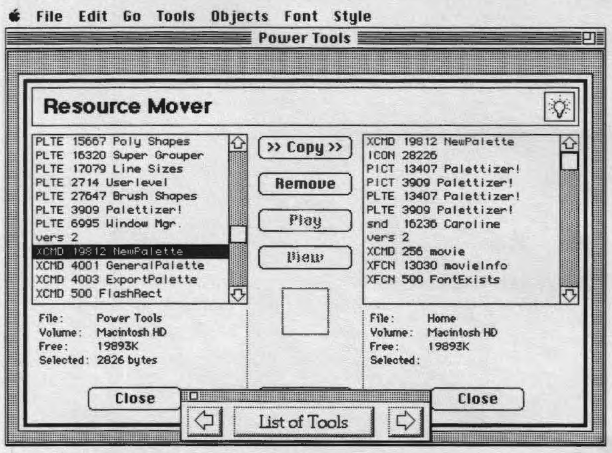
Begin by clicking the right arrow icon in the Power Tools stack's navigation palette until you see the Resource Mover card. If you prefer, you can click the List of Tools button, and then select Resource Mover from the list of tools available in the Power Tools stack.

Next, click the Open... button below the empty list box on the left side of the Resource Mover card. When HyperCard presents the Display Resources Of Which Stack dialog box, locate and double-click on the Power Tools stack. Now, click the Open... button below the list box on the right side of the card. This time, locate and open the Home stack.

Next, scroll through the Power Tools stack's resource list until you locate the `newPalette` XCMD. Click `newPalette` to select it. Then, click the Copy button to move the resource, as shown in Figure F on the next page.

After HyperCard installs the XCMD, it presents a dialog box telling you to quit HyperCard and relaunch it in order to use the new resource. To do so, simply click the dialog box's Quit button.

Figure F



```
PLTE 15667 Poly Shapes
PLTE 16320 Super Grouper
PLTE 17079 Line Sizes
PLTE 2714 Userlevel
PLTE 27647 Brush Shapes
PLTE 3909 Palettizer!
PLTE 6995 Window Mgr.
vers 2
XCMD 19812 NewPalette
XCMD 4001 GeneralPalette
XCMD 4003 ExportPalette
XCMD 500 FlashRect
```

```
File:     Power Tools
Volume:   Macintosh HD
Free:     19893K
Selected: 2826 bytes
```

```
XCMD 19812 NewPalette
ICON 28226
PICT 13407 Palettizer!
PICT 3909 Palettizer!
PLTE 13407 Palettizer!
PLTE 3909 Palettizer!
snd  16236 Caroline
vers 2
XCMD 256 movie
XFCN 13030 movieInfo
XFCN 500 FontExists
```

```
File:     Home
Volume:   Macintosh HD
Free:     19893K
Selected:
```

Use the Resource Mover to install the **`newPalette`** *resource in the Home stack.*

You now have all the resources you need to palettize a text field. After you reopen HyperCard, you'll be able to summon and use the Palettizer! palette. Now, let's take a look at how you can put the Palettizer! palette to work in your stacks.

Using the Palettizer! palette

It's easy to use the Palettizer! palette—you simply click on the field you want to palettize and then click the Palettize it! button on the Palettizer! palette. To demonstrate, first create a new stack named *Test Palette*.

Next, choose New Field from the Objects menu. Drag the new field to the upper-left corner of the card. Then, choose Background from the Edit menu to switch to the stack's background. Again, choose New Field from the Objects menu. Finally, choose Background from the Edit menu again to return to the card level. Choose the Browse tool from the Tools palette when you finish.

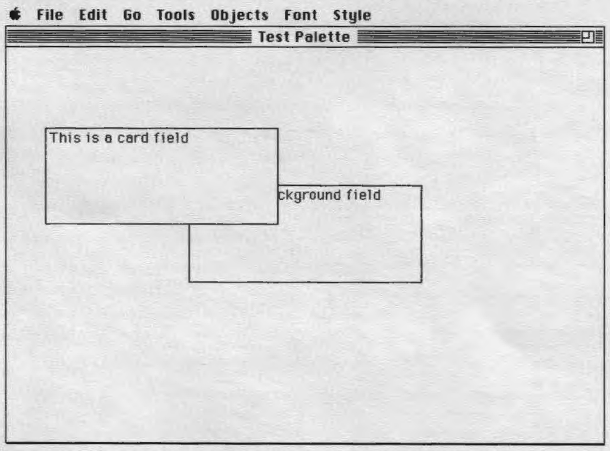
Now, click on the card level field (the first field you created) and type *This is a card field*. Then, click on the background field and type *This is a background field*. Choose the Field tool from the Tools palette and drag the card field so that it overlaps the text of the background field. Choose the Browse tool when you finish. At this point, your card should look like the one shown in Figure G.

Now you're ready to use the Palettizer! palette. To summon the palette, open the Message box and type

```
palette "Palettizer!"
```

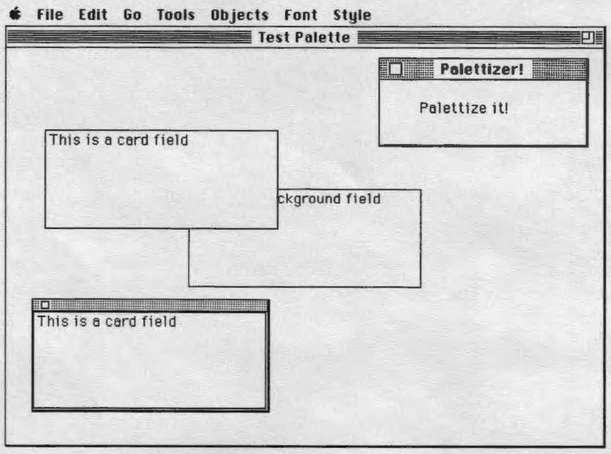
HyperCard will display the Palettizer! palette over the current stack. Next, click the card field to select it. Then, click the Palettize it! button. As Figure H shows, HyperCard creates a palette that contains a copy of the field.

Figure G



Drag the card field so that it overlaps the background field.

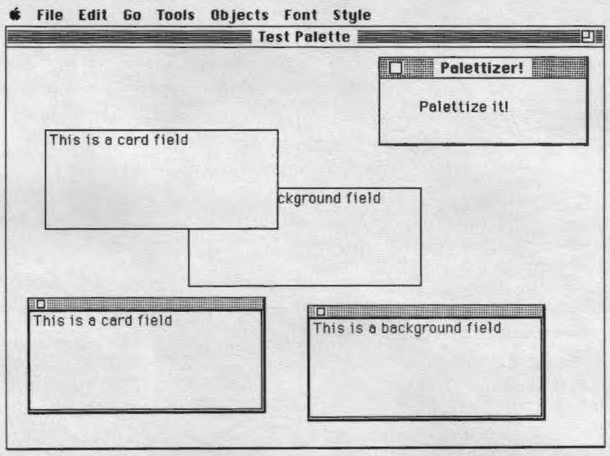
Figure H



HyperCard palettizes the selected field.

Next, click on the background field, and then click the Palettize it! button. As you can see in Figure I, HyperCard creates a palette that shows the text of the background field, even though the card field partially obscures it.

Figure I



HyperCard creates a palette that shows the entire background field.

Since the Palettizer! palette and its handlers exist in the Home stack, you can access them from any other HyperCard stack. You can reposition and close the palettes just as you'd close any other open window.

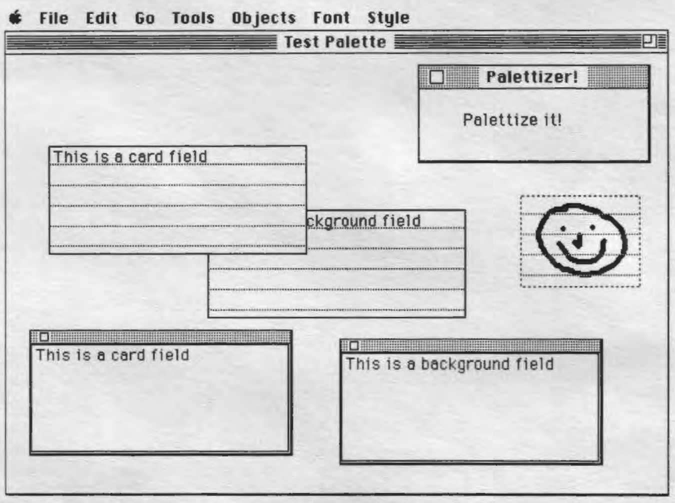
Another use for the Palettizer! palette

Because the Palettizer! palette and its handlers simply use the coordinates of the field you select to copy an area of a bitmap representation of your card, you can use the Palettizer! palette to place card or background artwork onto a palette. The trick to this technique is to create a transparent field and then place that field over the artwork you want to copy.

For example, choose the Brush tool and draw a happy face like the one shown in Figure J. Choose the browse tool from the Tools palette and then choose the New Field command from the Objects menu. Double-click on the new field to open its Field Info dialog box. Next, choose Transparent from the Style options. Click OK or press [return] to dismiss the Field Info dialog box.

When you return to the card, drag the new field over the happy face. Resize the field so that it exactly covers the face, as shown in Figure J. Choose the Browse tool when you finish.

Figure J



Resize the field so that it covers the happy face.

To palettize the happy face, simply click on it to select the transparent field surrounding it. Then, click the Palettize It! button on the Palettizer! palette. As Figure K shows, HyperCard palettizes the happy face.

You can use this technique to combine text and graphics on a palette, as well. You simply enter text into the transparent field after placing it over the artwork. Then, HyperCard will palettize both the text and the artwork.

Figure K

HyperCard palettizes the artwork beneath the transparent field.

Resource management

When you create a palette with the Palettizer! palette, HyperCard adds two resources to your stack's resource fork: a PLTE resource and a PICT resource, which govern the palette's appearance and function. HyperCard assigns these resources the same name as the field you're palettizing.

For example, the three palettes we created in our example added a total of six resources to the Test Palette stack. HyperCard created both PICT and PLTE resources named card field 1, bkgnd field 1, and card field 2.

To summon a palettized field not currently displayed on the desktop, you simply issue the palette command followed by the name of the palette resource. For example, to summon the first palette we created, you'd issue the command

```
palette "card field 1"
```

When you create a palette, HyperCard also generates a MacPaint file in the same folder as the open stack. As with the palette resources, this file has the same name as the field you're palettizing. The `palettizeIt` handler generates this file when it issues the `export paint to file` command.

For example, we generated the MacPaint files card field 1, bkgnd field 1, and card field 2 in the previous examples. After the handler imports these files, you no longer need to save them on your system. Unfortunately, you can't delete non-HyperCard files from within HyperCard unless you create an XCMD to do so. For this reason, you'll have to delete these files manually by dragging them into the trash.

If you don't delete these files, they will cause conflicts when you use Palettizer! in the future. The `export paint to file` command can't overwrite old MacPaint files when you create palettes from fields with the same names as fields you palettized earlier.

Conclusion

HyperCard lets you create cards that display text and graphics. However, you may occasionally want to move text or graphics from a card onto a palette. In this article, we showed you how to create the Palettizer! palette that allows you to generate custom palettes containing text or graphics. ■

Exporting and importing a snapshot of your stack

HyperCard provides a number of tools and commands designed to help you manipulate the graphic elements of your stacks. However, you may occasionally want to use another painting program to customize your stack's appearance. Although you can cut and paste graphic elements from one program to another, HyperCard provides an easy way to export a bitmap image of a card.

In this article, we'll discuss how you can use HyperCard's Import Paint... and Export Paint... commands. Along the way, we'll show you how to incorporate these commands into your scripts with the `import paint from file` and `export paint to file` commands.

How Import Paint... and Export Paint... work

The idea behind the Import Paint... and Export Paint... commands is simple. When you issue the Export Paint... command, HyperCard makes a bitmap image of the card. The image includes the visible portions of any fields or buttons on the card. The image also includes any visible background objects.

HyperCard exports the bitmap to a MacPaint file. You can open and edit this file using MacPaint or any other painting program that supports the MacPaint (MPNT) file format.

The Import Paint... command allows you to import MacPaint files (or files saved in the MacPaint format) into HyperCard. When you import a MacPaint file, HyperCard places that file as a single graphic that replaces the existing card picture.

The Import Paint... and Export Paint... commands both generate dialog boxes similar to the Open File or Save A Copy dialog boxes. In these dialog boxes, you can choose the file you want to import or name and save the file you want to export.

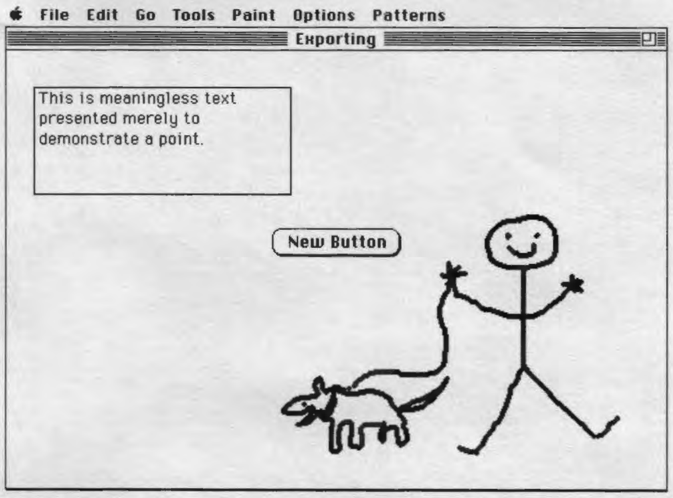
The Import Paint... and Export Paint... commands appear on the File menu only after you select one of the painting or selection tools. Similarly, you can use the `import paint from file` and `export paint to file` commands only when a painting or selection tool is active. Now, let's use the Import Paint... and Export Paint... commands to create a bitmap image of a card.

Taking the snapshot

Begin by opening a new stack. Next, choose the New Field command from the Objects menu. Drag the new field to the upper-left corner of the card. Then, choose New Button from the Objects menu to create a card button.

Now, choose the Browse tool from the Tools palette. Click on the field and type *This is meaningless text presented merely to demonstrate a point*. Finally, choose the Brush tool from the Tools palette. Use the brush to draw a simple graphic on the card. When you finish, your card will include a field, a button, and some card-level artwork, as shown in Figure A.

Figure A

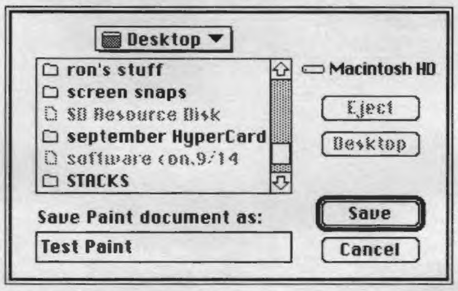


This is meaningless text presented merely to demonstrate a point.

Create a card containing a field, a button, and some artwork.

Now, choose the Export Paint... command from the File menu. When HyperCard presents the Export Paint dialog box, click the Desktop button (or use the navigation list box to navigate to the desktop if you're running under System 6). Then, type *Test Paint* into the Save Paint Document As text box, as shown in Figure B. Click Save or press [return] to export the card picture.

Figure B



Save Paint document as: Test Paint

Type the name Test Paint *into the Export Paint dialog box.*

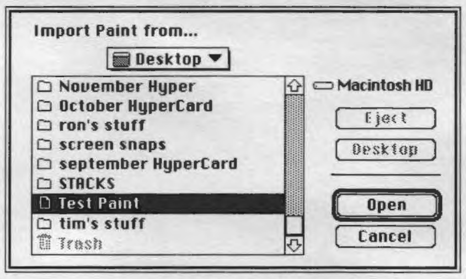
HyperCard will generate a MacPaint file named Test Paint on your Macintosh desktop. If you have MacPaint installed on your system, the icon will look like a standard MacPaint document icon. Otherwise, it will look like a generic document icon (▨).

Retrieving the snapshot

After you export a paint file, you can modify it with MacPaint or any other painting program that supports the MacPaint format. However, when you save the modified version of the file, you must save it in the MacPaint format.

Before you import the file, choose New Card from the Edit menu or press ⌘-N. Then, choose Import Paint… from the File menu. Navigate to the desktop in the Import Paint dialog box. Then, select the file Test Paint from the navigation list box, as shown in Figure C.

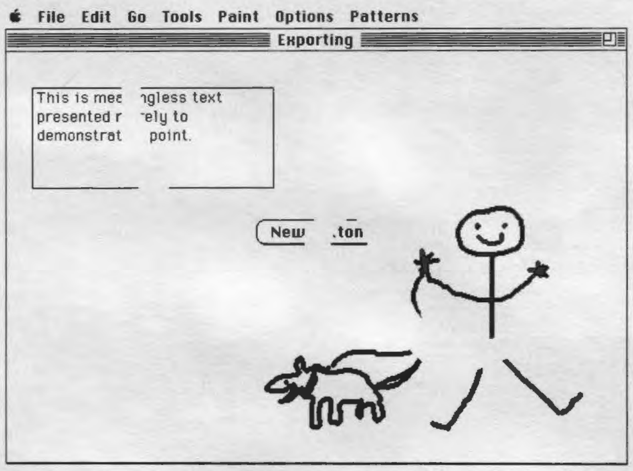
Figure C



Choose Test Paint from the Import Paint dialog box.

Click Open or press [return] to import the file. The imported card looks exactly like the original. However, the button and field are now only non-functioning bit-map copies of the originals. To demonstrate this, choose the Eraser tool from the Tools palette. Then, drag the eraser through the images of the field and button. As Figure D shows, you can erase the field and button images just like you'd erase other card-level artwork.

Figure D



The imported file contains non-functioning bitmap copies of the original field and button.

Automating the process

You can use the `import paint from file` and `export paint to file` commands to automate the process of importing and exporting paint files. These commands follow the syntax:

```
import paint from file "filename"
export paint to file "filename"
```

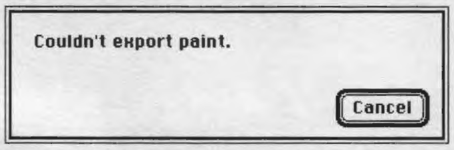
You must remember to choose one of the selection or painting tools before issuing either command. Otherwise, HyperCard will generate an error message.

If you make an error when you use the `import paint from file` or `export paint to file` command from the Message box, HyperCard will display the error message in a dialog box. For example, choose the Browse tool from the Tools palette. Then, enter the command

```
export paint to file "Painting"
```

Since you didn't choose one of the painting or selection tools, HyperCard generates the error message shown in Figure E.

Figure E



Couldn't export paint.

HyperCard generates this error message when you try to export the card picture with the Browse tool selected.

However, if you use the `import paint from file` and `export paint to file` commands within a script, HyperCard won't generate the error dialog box. Instead, it will continue with your script without executing the command. Fortunately, you can use the `result` function to identify an error before continuing with your script.

If the `export paint to file` command generates an error within a script, it sends the text string

```
Couldn't export paint.
```

to the `result` function. An importing error sends the text string

```
Couldn't import paint.
```

To trap the error messages in your scripts, you simply include the lines

```
if the result =¬
"Couldn't export paint." then
  ERROR TRAPPING CODE
end if
```

Of course, you'd supply the string

```
Couldn't import paint.
```

if you were trying to trap an importing error.

Finally, if you attempt to export a card image to a filename that already exists, HyperCard will generate the

```
Couldn't export paint.
```

error message. For example, if you issue the command

```
export paint to file "aardvark"
```

twice, HyperCard will present the error dialog box the second time you execute the command. To overwrite an existing MacPaint file, you must use the Export Paint… command from the File menu.

LETTERS

A rounding oddity

The article "Rounding to the Next Higher or Lower Whole Number," in the July 1993 issue of *Inside HyperCard*, isn't quite complete in its description of how HyperCard's `round()` function works. Because HyperCard uses an engineering convention when it rounds numbers, the `round()` function may not always return the numbers you expect.

Your article states

"If the decimal portion of a number is less than .5, `round()` returns the next lower whole number; otherwise, it returns the next higher whole number. In other words, the statement

```
round(5.5)
```

returns the value 6."

Other readers may be surprised to find that if they enter the command

```
round(6.5)
```

HyperCard again returns the value 6. This seems a bit confusing, and it contradicts the statement you made in the article.

HyperCard follows a convention for rounding columns of values for statistical or engineering purposes that I learned in engineering school. Consider that there are nine possible values for the tenths place in a decimal number—.1, .2, .3, .4, .5, .6, .7, .8, and .9. If you round all decimal values less than .5 to the next lower number, you potentially round down in four cases (.1, .2, .3, .4) and round up in five cases (.5, .6, .7, .8, .9).

To balance this discrepancy, you can follow a simple convention regarding decimal values of .5. If the integer portion of a number is even, then the decimal will round down. If the integer portion is odd, then the decimal will round up.

Consequently, the statement

```
round(118.5)
```

returns `118`, while the statement

```
round(119.5)
```

returns `120`.

Since this isn't the result most lay people expect, you may want to take precautions when using the `round()` function in your stacks.

Lou Smith
Succasunna, New Jersey

Mr. Smith is correct in his observation about HyperCard's `round()` function. We tested several Macintosh and PC applications and found that approximately half follow this convention. In fact, some software companies follow this convention in one or two of their products, but not in others.

Mr. Smith also sent a suggestion for those scripters who would like HyperCard to employ the more conventional (but less accurate) method of rounding. You can create a user-defined function to round numbers in the more traditional method.

To do so, open your Home stack. Then, choose Stack Info from the Objects menu. Next, click Script… to open the stack script. Scroll to the Utility Scripts section of the stack script. Then, enter the following function handler:

```
function newRound val
  if trunc(val) mod 2 = 0 then
    if val-trunc(val)≥ .5 then
      put trunc(val)+1 into val
    end if
  else
    put round(val) into val
  end if
  return val
end newRound
```

Save the script and close the Script Editor.

To test this function, open the Message box and type

```
newRound(5.5)
```

When you press [return], HyperCard returns the value 6.

Now, enter the statement

```
newRound(6.5)
```

This time, the `newRound()` function returns the value 7, instead of the value 6 the `round()` function returned.

As always, we're grateful for your observations and suggestions. To thank Mr. Smith for his contribution, we're sending him a check for $25. ■

Printed in the USA
Printed on recycled and recyclable paper
50% recycled fiber, including 10% post-consumer waste paper